

A Server-Side System for the Analysis of AC Interference on Pipelines



**BACHELOR OF ENGINEERING PROGRAM IN SOFTWARE ENGINEERING
INTERNATIONAL COLLEGE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2017
KMUTL-2018-IC-B-003-003**

Thesis - Academic Year 2017

Bachelor of Engineering in Software Engineering International College
King Mongkut's Institute of Technology Ladkrabang

Title A Server-Side System for the Analysis of AC Interference on Pipelines

Authors Mr. Polawat Nakthong

57090026

Mr. Yosavich Daolomchan

57090044



(Dipl.Ing Michael Juelich)

Project Advisor

Date 11 / 06 / 2018

A blue ink signature of Dr. Natthapong Jungteerapanich.

(Dr. Natthapong Jungteerapanich)

Project Co-Advisor

Date 18 / 6 / 2018

Abstract

Polawat Nakthong 5790011

Yosavich Daolomchan 57090044

Dipl.Ing Michael Juelich Advisor

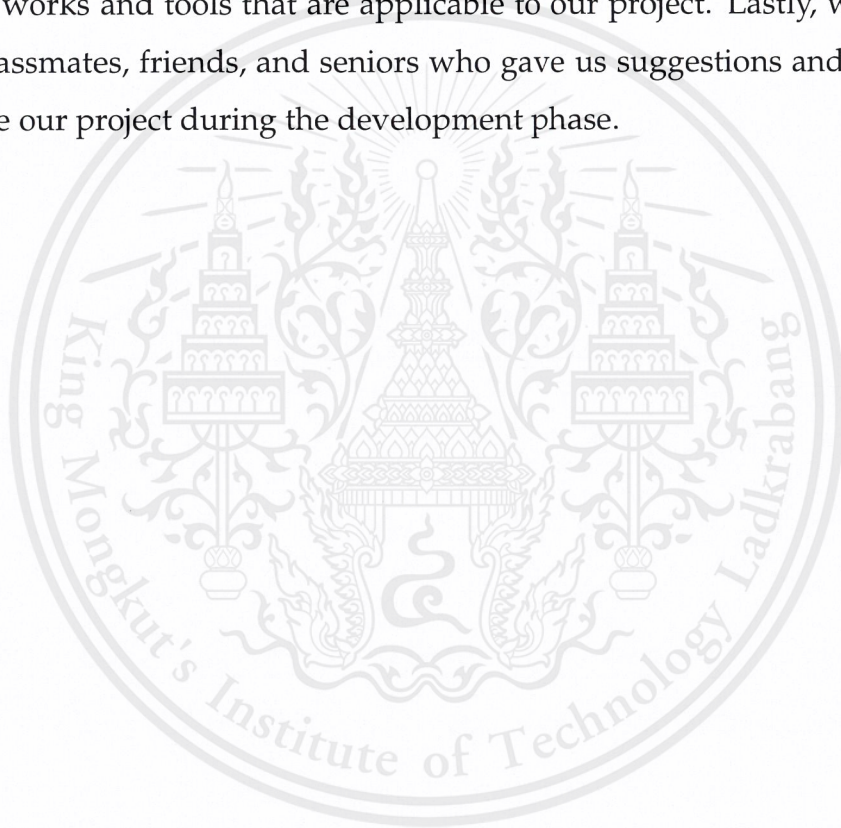
Academic Year 2017

When pipelines are installed near high-voltage AC power lines, a significant amount of voltage will be induced on the pipelines. Such a voltage interference can damage the protection system which was installed on the pipelines to prevent corrosion, thus potentially rendering the pipelines corrosion-prone. It can also harm the personnel working near the pipelines. The intention of this project is to be a first step toward a software application that can be used as a workbench to support the analysis of the interference effects of high voltage AC from power lines on nearby pipelines and assist the user in designing a suitable solution to protect the pipelines.

The scope of this project is to develop the back-end of a Geographic Information System (GIS) to provide 2-dimensional maps, pipeline positions and other related geographical objects which will be used by the client-side web application for visualizing geospatial data. The geospatial data service is running on an Apache HTTP server and using Django. For the tile map service (TMS), we use Mapnik to render tile map images from geospatial data stored in the PostGIS database. The client-side web application can communicate with the server-side system through HTTP protocol.

Acknowledgments

We would like to thank Dipl. Michael Juelich, our project advisor, for giving us a great opportunity to work in a real-world project and providing us the necessary information for our project. With the communication, translation problems and most important are teaching us an electrical engineering knowledge which we we faced during the project development and also in providing us technical details and documents needed in our project. We would also like to thank Dr. Natthapong Jungteerapanich, our project co-advisor, for his advice on different programming techniques and development methodologies, as well as related works and tools that are applicable to our project. Lastly, we would like to thank our classmates, friends, and seniors who gave us suggestions and ideas so that helped improve our project during the development phase.

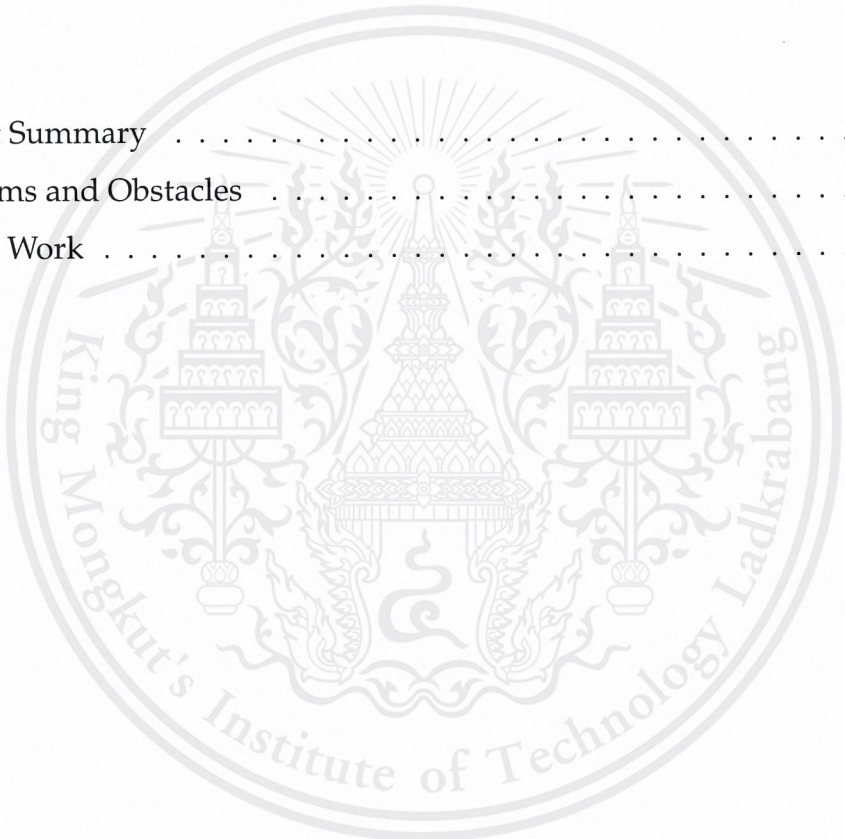


Contents

1	Introduction	9
1.1	Problem Description	9
1.2	Motivation	12
1.3	Objectives	12
1.4	Scope of Work	13
2	Background Knowledge and Related Works	14
2.1	Background Knowledge	14
2.1.1	Calculations of electrical interference on pipelines	14
2.1.2	Induced slow-field strength	17
2.1.3	Magnitude wave characteristic and decisive characteristic	18
2.1.4	Pipeline segmentation	18
2.2	Geographic Information System	19
2.2.1	Coordinate system	19
2.2.2	Map projection	20
2.2.3	Earth datum	22
2.2.4	Distance between two locations	23
2.2.5	Closest point on the line	24
2.3	Review of Related Works	24
3	Requirements Analysis	26
3.1	System Overview	26
3.2	System Requirements	27

3.2.1	Functional requirements	27
3.2.2	Non-functional requirements	28
3.3	Use Case Diagram	29
4	Software Design	31
4.1	Software Tools	32
4.1.1	GeoDjango	32
4.1.2	Mapnik	32
4.1.3	PostGIS	32
4.2	Services	32
4.2.1	Get user authentication	32
4.2.2	Tile map service	33
4.2.3	Calculation service	33
4.2.4	Search service	33
4.2.5	Get geographical object service	34
4.2.6	Create geographical object service	34
4.2.7	Edit geographical object service	35
5	Development	36
5.1	GIS Data Formats	36
5.1.1	Vector data	36
5.2	Database	37
5.2.1	Spatial database system	37
5.2.2	Importing geospatial data into a spatial database	37
5.3	Geographical Object Service	38
5.4	Tile Map Rendering	39
5.5	Electrical Interference Calculation	40
5.6	User Authentication and Registration System	40
5.6.1	Registration Service	41
5.6.2	Authentication of Services	41

6	Results	43
6.1	Tile Map Service	43
6.2	Registration	44
6.3	Token Authentication	46
6.4	Geographical Object Services	46
6.4.1	Sample of the description of a power line in the GeoJSON format . .	47
6.4.2	Sample of the description of a segmented line in the GeoJSON format	47
6.4.3	Sample of the description of a pipeline in the GeoJSON format . . .	48
6.5	Calculation	49
7	Conclusion	51
7.1	Project Summary	51
7.2	Problems and Obstacles	51
7.3	Future Work	52



List of Figures

1-1	Pipeline coatings and linings	10
1-2	Cathodic protection system	10
1-3	Induced current in pipeline	11
2-1	Pipeline potential diagram[2]	15
2-2	Graphs of U_R and I_R values [2]	16
2-3	Graph of Mutual inductance (M') according to the distance between the pipeline and the power line [2]	17
2-4	Pipeline segmentation	19
2-5	Example of GIS data layers [12]	20
2-6	Cylindrical projection method [12]	21
2-7	Example of Mercator projection map [12]	21
2-8	Earth's surface [13]	22
2-9	Example of Earth datum[13]	23
2-10	The point on a line that is closest to the given point	24
3-1	Initial design of the system architecture	26
3-2	Use case diagram	29
4-1	Software design of our application	31
6-1	A map generated by Mapnik	44
6-2	Password is sent via e-mail	45
6-3	Administrator can change status of user	45

Chapter 1

Introduction

1.1 Problem Description

Steel pipelines that are commonly used to transport petroleum products, such as oil or natural gas, are prone to corrosion. Two popular and effective solutions to this corrosion problem are

1. Protective coatings and linings
2. Cathodic protection (CP)

Coatings and linings, both internal and external, are a basic method of defending the pipeline against the corrosion. The internal lining provides protection against gasoline, crude oil, and natural gases. The external coating protects the pipeline against corrosion from the surrounding environment. Figure 1-1 shows an example of protective coating layers.

A cathodic protection system (CP) is a technique used to control and decrease the corrosion rate of the pipe's metal surface by making it the cathode of an electrochemical cell. It involves injecting DC current into the pipeline.

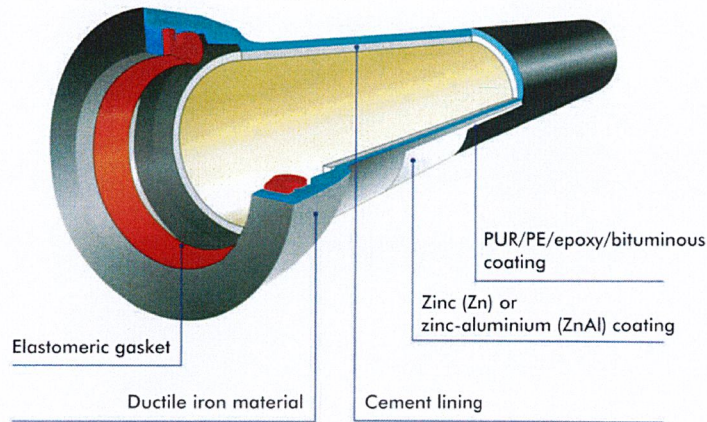


Figure 1-1: Pipeline coatings and linings

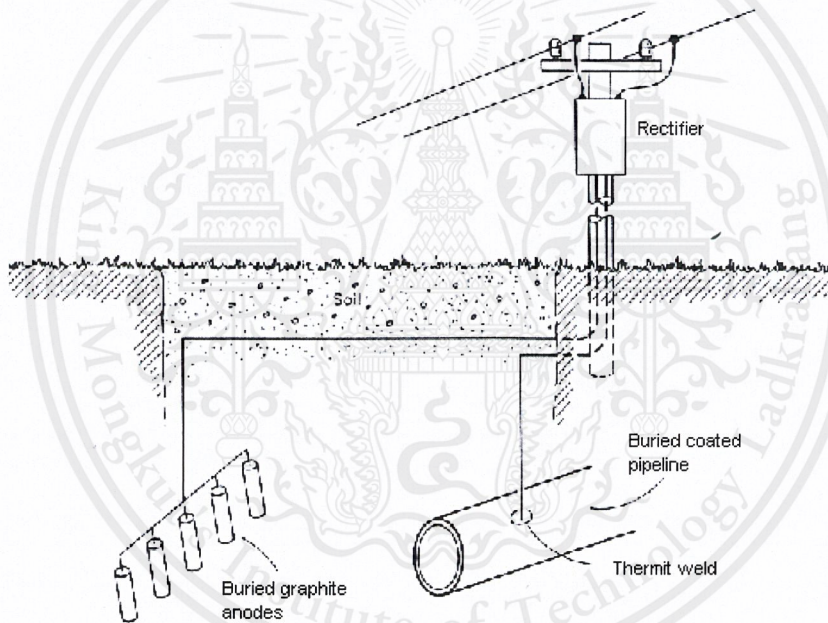


Figure 1-2: Cathodic protection system

When a steel pipeline runs close to a high-voltage AC power line, the AC current on the power line can induce an AC current on the pipeline. If the induced AC current on the pipeline gets above a certain level, it can cause the following problems:

1. The induced current on the pipeline can pose safety hazard to the person working in contact with or near the pipeline.

2. The induced current can damage the protective coating and lining of the pipeline.
3. The induced current can damage the cathodic protection system, for example, the DC power source, that is equipped on the pipeline.
4. The induced current can damage other equipments installed next to the pipeline.

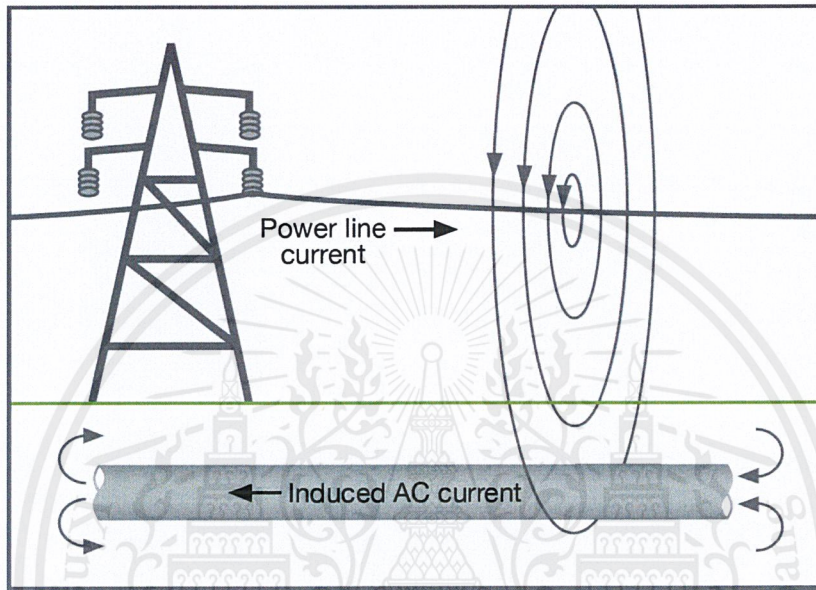


Figure 1-3: Induced current in pipeline

To prevent these problems, additional protection devices can be installed on the pipeline to reduce the detrimental effect of the interference from the nearby power line. Since a pipeline network is vast and typically only some parts of the network are potentially affected by the AC interference effect, the company operating the pipeline network needs to determine which sections of the network require protection. This involves complex calculations using various information on the pipeline network, the power-line network, and the surrounding environment. For a large pipeline network, this is time-consuming and prone to errors. This has led us to develop a software application to assist the user in performing the calculations to determine the extent of the AC interference effect on each section of a pipeline network.

1.2 Motivation

Pipeline transportation is one of the ways to transport goods or material through a pipe. Pipeline transport has also played an important role in the transmission of petroleum products because pipelines are still the cheapest and most efficient way to transport oil and natural gas. A pipeline is able to transmit 700,000 barrels per day so, to be able to transport all the product safely, every pipeline needs to be checked and monitored and equipped with a corrosion protection to prevent it from corrosion. Damaged pipelines could cause a lot of damage to the environment, people and properties nearby. However, the protection system may be damaged when it is affected by an interference from the high voltage AC current on a nearby power line.

To prevent the corrosion protection system from AC interference, an additional protection system needs to be installed. In the worst case, the corrosion problem can occur anywhere along the pipeline. The interference problem can happen only when the power lines and the pipelines are installed close to each other. So, to install an interference protection all the way of the pipeline would be unnecessary and also a waste of money.

To solve this problem, a pipeline company needs to analyze the pipeline and power line then design an interference protection system. The company also has another option such as hiring an outsource company to do the analysis, but currently there are not many organizations in the world that could offer this kind of services for the fact that it takes specialized knowledge and a long time. The most important thing is that it costs much money.

1.3 Objectives

To build the back-end for a web-based application that facilitates the analysis of electrical interferences from a power line network on a pipeline network which can be used in the real situation.

1.4 Scope of Work

In order to develop server-side application, we need to:

1. Set up Django with the Geojango extension as a web framework and middleware between the client-side application and the server-side system.
2. Set up PostgreSQL with the PostGIS extension which we will use for spatial data manipulations.
3. Implement and develop a Django GIS application to integrate each system component together and provide REST services to the client-side application.



Chapter 2

Background Knowledge and Related Works

2.1 Background Knowledge

2.1.1 Calculations of electrical interference on pipelines

To simplify the analysis, we have made the following assumptions on the pipeline and power line configuration.

In order to analyze the electrical interference and the shielding area throughout the pipeline path, we make some assumptions.

1. The pipeline runs parallel to the influencing power line.
2. The coordinate designation is such that $x = 0$ at the center of the power line and $x = \pm \frac{l}{2}$ at the end of the pipeline.
3. The pipeline is terminated on both sides with the shaft resistor.
4. The pipeline is continuously grounded.
5. The pipeline has a uniform resistive resistance.
6. The pipeline has a specific ground resistance along the parallel path.

7. In the case of short-term influence, the earth (ground) short-circuit occurs outside the parallel line.

Figure 2-1 shows a circuit diagram for the electrical potentials and current on the pipeline that is influenced by an external electromagnetic field.

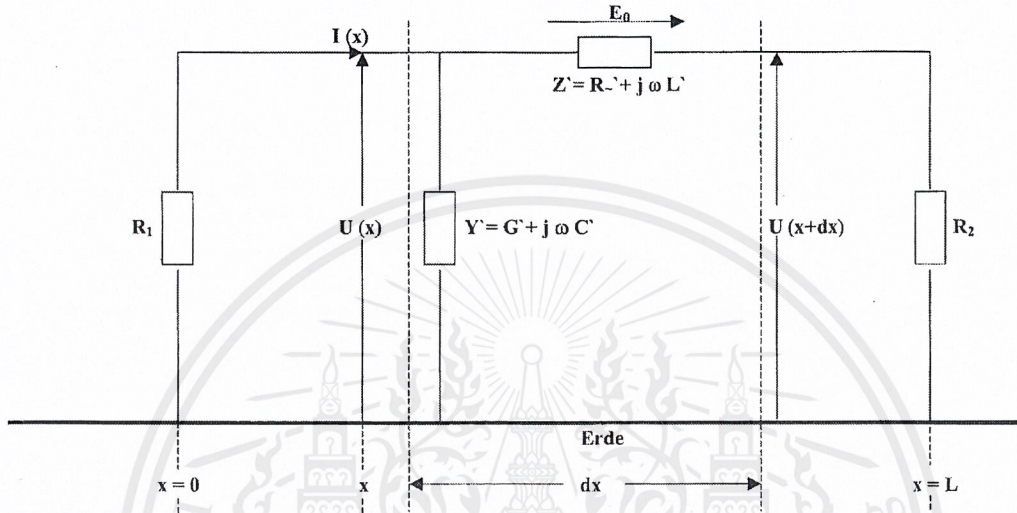


Figure 2-1: Pipeline potential diagram[2]

From the circuit diagram in Figure 2-1, the induced voltage and current on the pipeline can be computed from the following equations [2]:

$$|U_R| = \frac{|E|}{2 \times |\gamma|} \times \frac{|e^{\gamma x}|}{|e^{\gamma \frac{L}{2}}|} \times |1 - e^{-2\gamma x}| \quad (2.1)$$

$$|I_R| = \frac{|E|}{|Z_W| \times |\gamma|} \times \left| \left(1 - \frac{\cosh(\gamma x)}{e^{\gamma \frac{L}{2}}} \right) \right| \quad (2.2)$$

where

- $|U_R|$ is the induced voltage in pipeline
- $|I_R|$ is the induced current in pipeline
- $|E|$ is the value of induced slow-field strength in the ideally insulated conductor.
- $|Z_W|$ is magnitude wave characteristic impedance of the pipeline.
- $|\gamma|$ is the amount of the transmission rate of the pipeline.
- γ is a complex value of transmission rate on the pipeline.
- L is length of pipeline that are installed in parallel direction to the pipeline.

Figure 2-2 shows an expected result of the calculation from Equation 2.1 and 2.2 when the pipeline diameter is 600 meters and the length of pipeline is 12 kilometers.

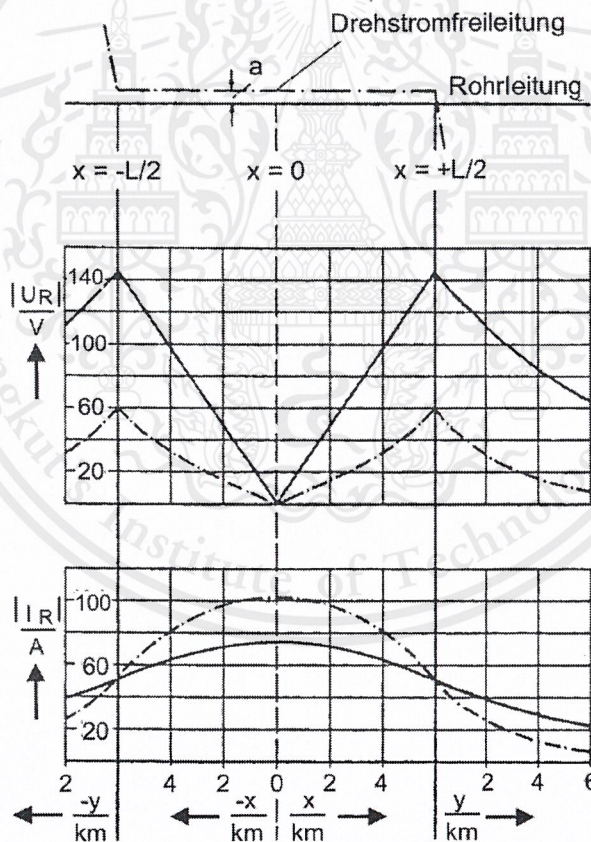


Figure 2-2: Graphs of U_R and I_R values [2]

2.1.2 Induced slow-field strength

The value of induced slow-field strength in an ideally insulated conductor shown in Equation 2.1 and Equation 2.2 is calculated from Equation 2.3.

$$E = 2\pi \times f \times M' \times l \times I \times r \quad (2.3)$$

where the

E = Value of slow-field strength

f = Frequency of the electricity transmitted on the power line

M' = Mutual inductance show in Figure 2-3

l = Length of the power line

I = Amount of current flow on the power line

r = Radius of the power line

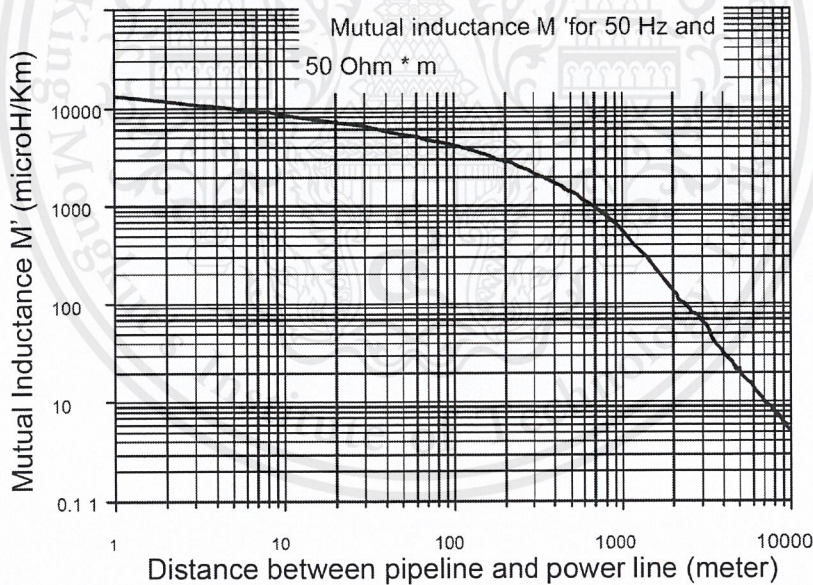


Figure 2-3: Graph of Mutual inductance (M') according to the distance between the pipeline and the power line [2]

2.1.3 Magnitude wave characteristic and decisive characteristic

The magnitude wave characteristic and decisive characteristic used in finding the electrical interference in Equation 2.1 and Equation 2.2 can be defined using Equation 2.4 and Equation 2.5.

$$\gamma = \sqrt{Z' + Y'} \quad (2.4)$$

$$Z_W = \sqrt{\frac{Z'}{Y'}} \quad (2.5)$$

where

- γ = Magnitude wave characteristic
- Z_W = Decisive characteristic
- Z' = complex impedance of the pipeline
- Y' = complex derivative admittance of the pipeline

The value of Z' and Y' are to be collected on-site and provided to the system by the utilities company.

2.1.4 Pipeline segmentation

As pipeline and power lines are installed over a long distance, it is likely that these lines contain joints within its path. Splitting a pipeline into sections which are then used to calculate the electrical interference dependently allows the analysis to be more accurate and reliable since the formulas are designed to be used only when the pipeline and the power line are installed in parallel.

A pipeline can be divided into sections, based on the locations of the joints on the pipeline and those on the power lines. If the distance between the pipeline and the nearby power line is larger than one kilometer, the section can be omitted and will not be included in the calculation as the electrical interference is very low and does not effect the pipeline protection system. An example of pipeline segmentation is shown in Figure

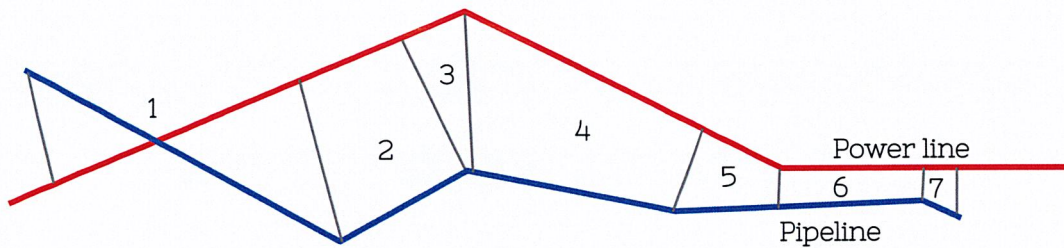


Figure 2-4: Pipeline segmentation

2.2 Geographic Information System

A Geographic Information System (GIS) is a computer-based system for mapping, capturing, and analyzing data related to positions on Earth's surface [12].

Many different types of information can be integrated into a GIS and represented as a map layer which can include streets, parcels, zoning, flood zones, landmarks, etc.

GIS often contains a large variety of data that do not appear in an on-screen or printed map. Figure 2-5 shows an example of data layers in GIS.

2.2.1 Coordinate system

A coordinate system is the system that uses one or more numbers to represent a point or position. When a position on Earth is marked, that position will be transferred to the map using the coordinate system. Two numbers represent a coordinate in a two-dimensional space and three numbers represent a coordinate in a three-dimensional space.

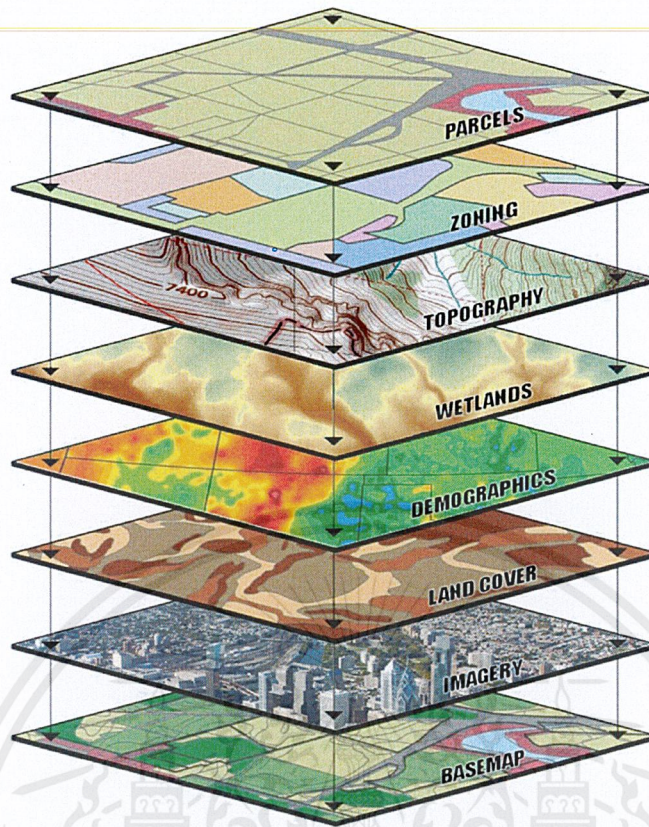


Figure 2-5: Example of GIS data layers [12]

2.2.2 Map projection

Projection is a transformation which changes 3-dimensional Earth geometry into a 2-dimensional map. There is no projection that is perfect. For each map, we need to choose a projection that suits our purpose best. In this project, we use Mercator Projection, which is a cylindrical map projection, because it is a standard for web mapping applications and is widely used [12]. Figure 2-6 demonstrates the cylindrical projection method. Figure 2-7 shows an example of a map that is obtained by the Mercator projection technique.

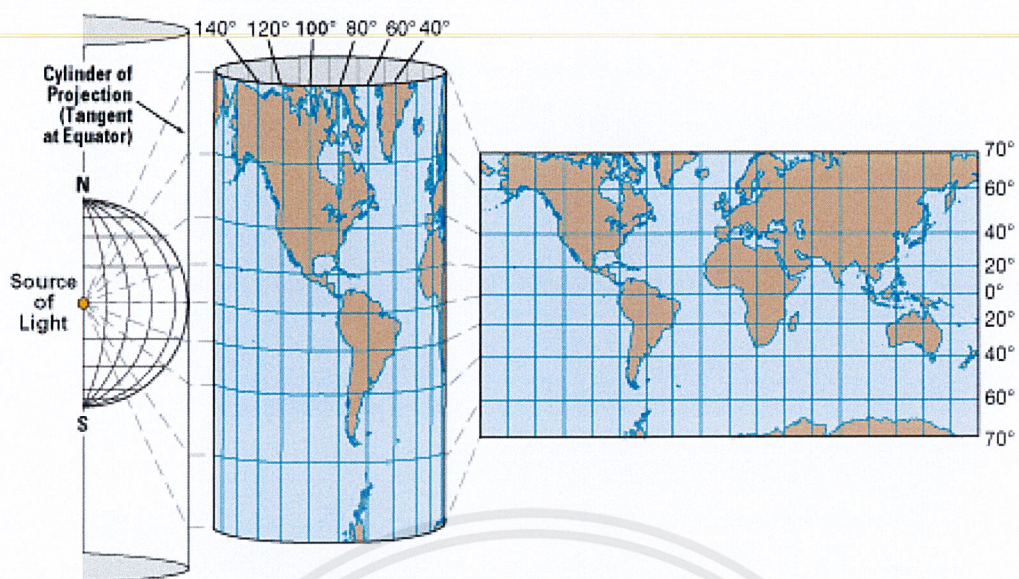


Figure 2-6: Cylindrical projection method [12]



Figure 2-7: Example of Mercator projection map [12]

2.2.3 Earth datum

Earth in reality is a very misshapen object. It is not really round. The shape of the earth is called ellipsoid. Datum is a model which contains a set of number which captures the shape and size of an ellipsoid. Each country or continent chooses a different datum to suit their location the best. The GPS network of the Royal Thai Survey Department and almost every nation use WGS 84 datum to determine the position for this reason we adopt the WGS 84 in our project. The latest standard revised in 2004 is used as its error is believed to be under two centimeters [13]. Figure 2-8 shows an example of the Earth surface. Figure 2-9 illustrates an Earth datum.

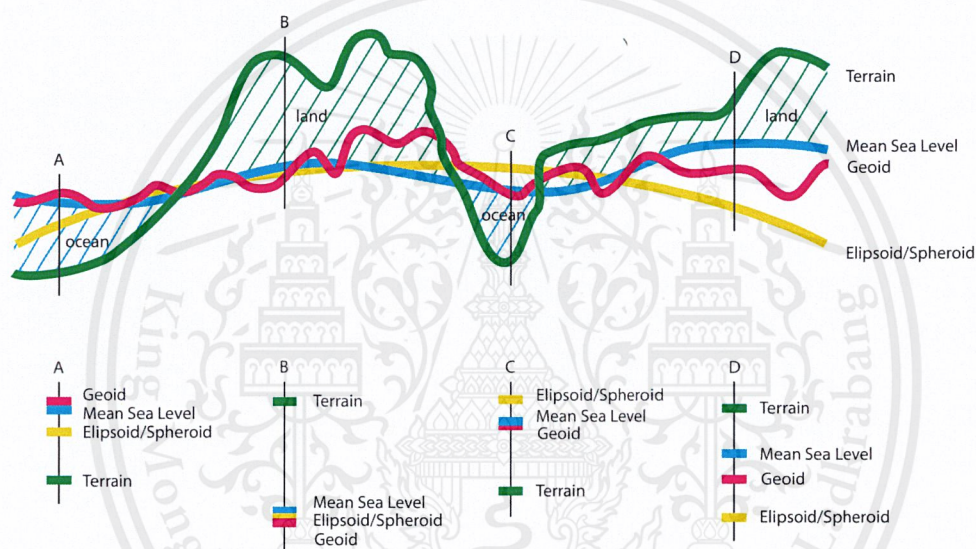


Figure 2-8: Earth's surface [13]

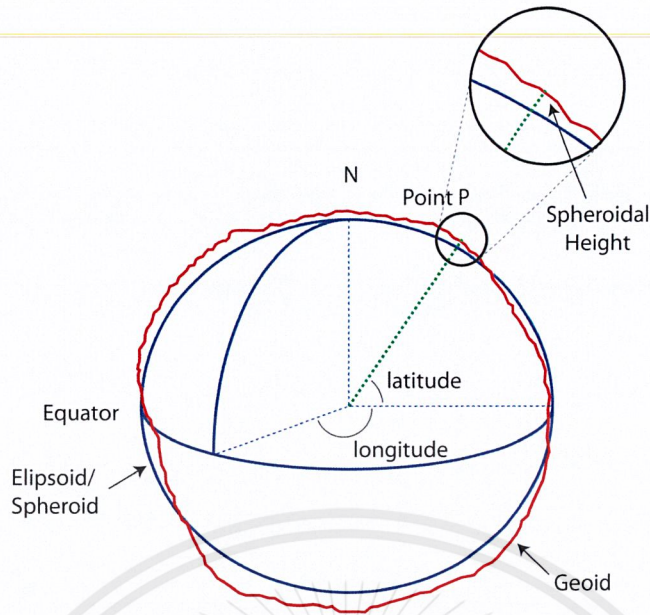


Figure 2-9: Example of Earth datum[13]

2.2.4 Distance between two locations

By implementing the law of haversine, we can calculate the accurate distance between two points on a spherical surface. The haversine formula determines the great-circle distance between two points on a sphere given by their longitudes and latitudes as shown in Formula 2.6.

$$D = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\sigma_2 - \sigma_1}{2} \right) + \cos \sigma_1 \cos \sigma_2 \sin^2 \left(\frac{\lambda_1 - \lambda_2}{2} \right)} \right) \quad (2.6)$$

where :

D is the distance between two points.

r is the radius of the sphere (in this case, the Earth).

σ_1, σ_2 are latitude of Point 1 and latitude of Point 2.

λ_1, λ_2 are longitude of Point 1 and longitude of Point 2.

2.2.5 Closest point on the line

To segment a pipeline, we need to find the point on the power line that is closest to each pipeline corner. The point is obtained by transforming the power line into a linear equation of the form $ax + by + c = 0$, then the values of a and b are substituted in Equation 2.7 the latitude value and Equation 2.8 for the longitude value.

$$x = \frac{b(bx_0 - ay_0) - ac}{a^2 + b^2} \quad (2.7)$$

$$y = \frac{a(-bx_0 + ay_0) - bc}{a^2 + b^2} \quad (2.8)$$

This method will result in some error due to the Earth's spherical shape. However, the error is acceptable since the concerned distance is under one kilometer. Figure 2-10 shows the diagram of the closest point (x_L, y_L) on the line found using the formulas above.

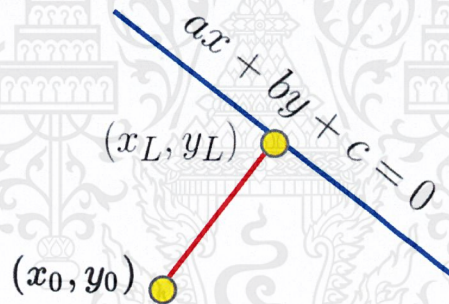


Figure 2-10: The point on a line that is closest to the given point

2.3 Review of Related Works

This project uses a spatial database to implement a GIS system. In this section, we will discuss some advantages and disadvantages of widely used spatial databases. There are two most popular spatial databases that we are going to discuss here, which are Oracle Spatial and PostGIS. In terms of performance, PostGIS accesses and queries spatial data faster than Oracle Spatial, with an average of 50 percent speed up and more than 4 times

speed up in the best case. Another advantage of PostGIS is its clear and detailed technical documents which can easily be found and widely used by GIS developers. With these clear advantages of PostGIS, we have decided to use it in our project.



Chapter 3

Requirements Analysis

3.1 System Overview

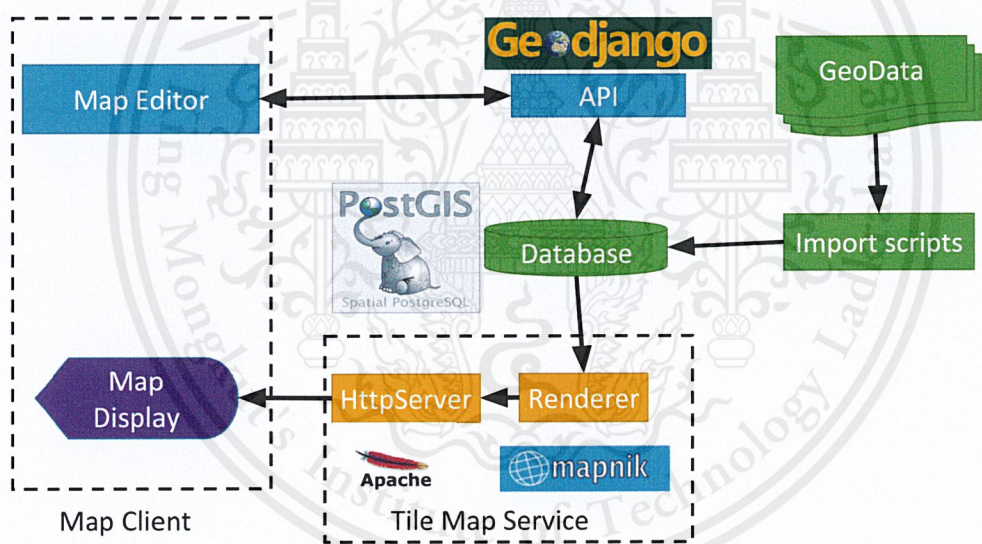


Figure 3-1: Initial design of the system architecture

Figure 3-1 shows the initial design of the system architecture of the server side of our workbench for the analysis of AC interference on pipelines.

PostGIS is the extension for the PostgreSQL database system which adds geospatial data types and functions that make handling and manipulating spatial data in a database

easier. Geographic data needs to be imported to our database. It consists of raster files and vector files, for example, GeoTIFF which is a raster image format that contains geo-reference data in each pixel of the image. For a vector file, we use the OSM xml format which records geographical data from the OpenStreetMap website in XML format.

We use Django with Geodjango extension as a middleware between the client-side application and the server-side system. When the user edits a map in the map client, the map client will receive and send data to the database using the Geodjango API. Geodjango API is a REST web service that can read from and write to the database. We use GeoJson file format for handling geospatial data. The networking protocol that the system uses for a connection between Client and API is HTTP.

For the rendering process, we use Mapnik to render data from the PostGIS database into tile map images in different zoom level.

3.2 System Requirements

3.2.1 Functional requirements

1. The system can import geographic data from both vector and raster files into the database.
2. The system can generate map tile images with various zoom levels from spatial data in the database.
3. The system provides different functionalities for each type of users, namely,
 - (a) registered users which include normal users, analyzers and administrators, and
 - (b) unregistered users.
4. An unregistered user can register to access the system.
5. A registered user can login and logout.

6. A registered user can request map tile images from a given coordinate and zoom level using an HTTP request.
7. A registered user can retrieve and view geographical objects from the system.
8. A registered user can search and locate geographical objects.
9. An analyzer can request the result of a calculation of the electrical interference on the specified pipeline and power line.
10. An analyzer can manipulate the properties of geographical objects in the system.
11. An administrator can access and edit data in the database.
12. An administrator can assign a user type for each registered user.

3.2.2 Non-functional requirements

1. The spatial data sources are in the form of vector data and raster data.
2. Geospatial data projection used in the system is EPSG:3857.
3. The system must be deployable on a Windows 10 64-bit platform.
4. Normally, the system must be running all the time.
5. The system must use the Mapnik module to generate tile map images.
6. The system must use Django with Geodjango extension as a web framework between the client application and the server system and a PostgreSQL database with the PostGIS extension installed.
7. The database in the system must guarantee data consistency.

3.3 Use Case Diagram

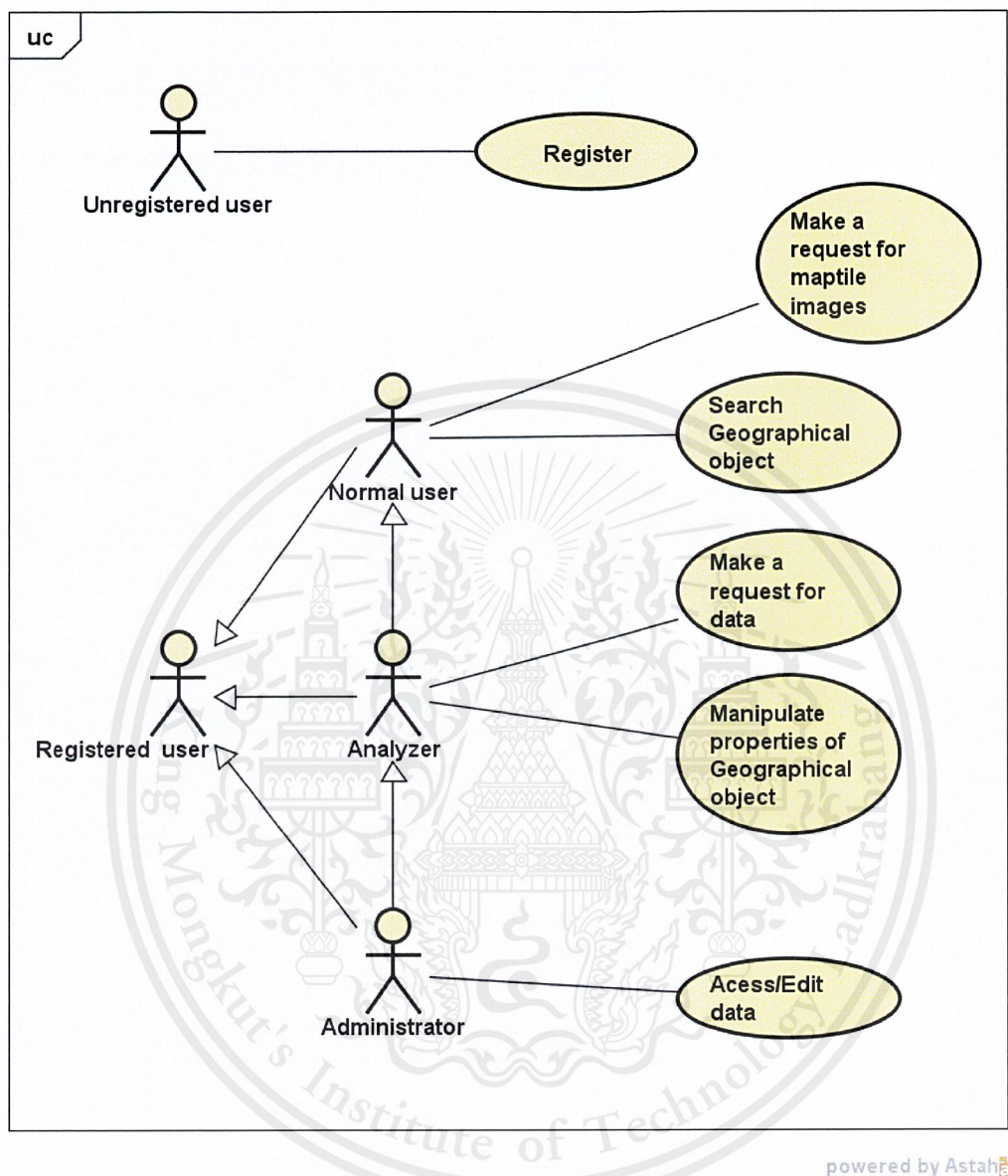


Figure 3-2: Use case diagram

Figure 3-2 shows the use case diagram for the server-side of our workbench for the analysis on AC interference on pipelines.

There are four actors, namely, unregistered user, registered user, administrator and analyzer. An unregistered user can register with the system in order to be a registered

user. A registered user can log into and log out of the system. After successfully logging in, a registered user can make a request for map tile images and search/locate geographical objects. The analyzer can do all the things a registered user can do and additionally making a request for mathematical data and manipulating the properties of a geographical object. The administrators can do everything that an analyzer can do and additionally accessing and editing data from the database.



Chapter 4

Software Design

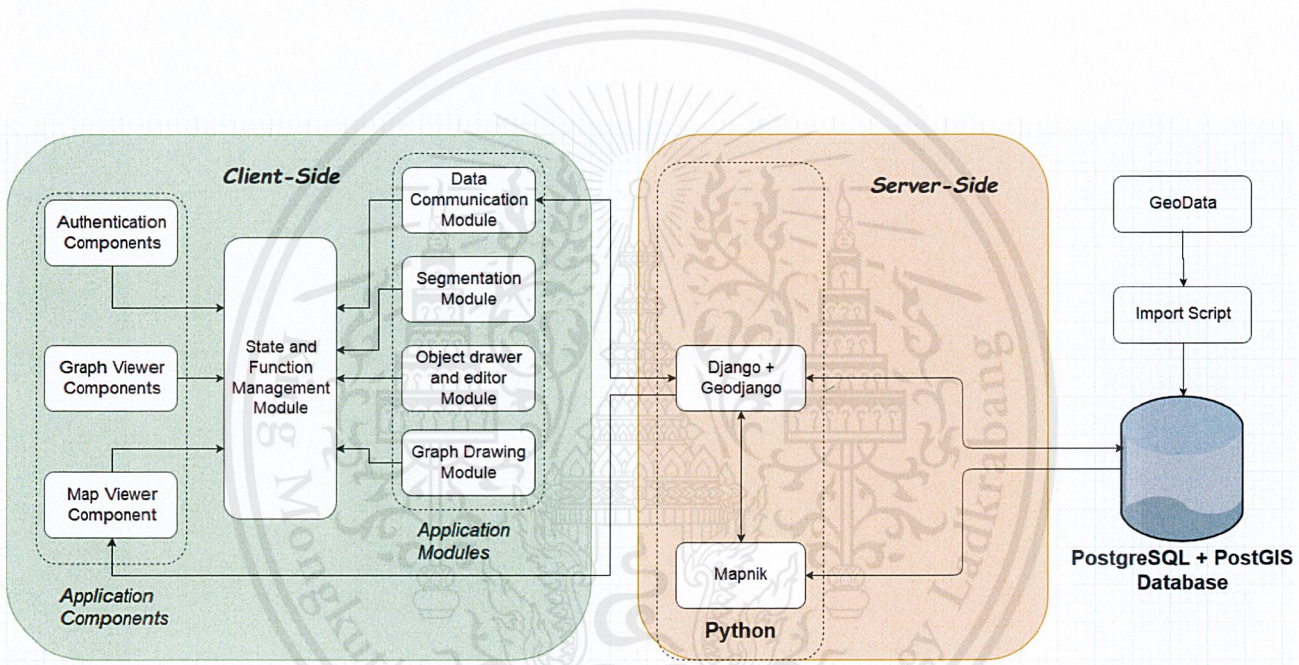


Figure 4-1: Software design of our application

Figure 4-1 shows the design of our application, which mainly consists of the client-side and the server-side. The server-side system architecture of our application has two main components, which are the server component and the database component. The server component consists of Mapnik and GeoDjango. The client and the server communicate using GeoJson data type.

4.1 Software Tools

4.1.1 GeoDjango

GeoDjango is an extension of Django. It is used to develop a GIS web application. It consists of some libraries which can be used to perform geospatial functions, such as manipulating spatial data in a PostGIS database and managing geographic data which use between the client-side application and the database. GeoDjango APIs allow creating, updating and deleting geographic features [15].

4.1.2 Mapnik

Mapnik is a C++ map rendering toolkit which can be scripted using Python. Mapnik is used for rendering map images from spatial data in a PostGIS database according to the x, y coordinate and the zoom level of each datum. An apache module called *mod_tile* is used to serve the map tile images created by Mapnik [16].

4.1.3 PostGIS

PostGIS is a spatial database extender. It adds geographical object support to PostgreSQL object-relational database by adding spatial data types and extra functions and operations for manipulating spatial data. The additional functions and operations can be used in PostgreSQL queries. PostGIS is a necessary core component that works with both Mapnik and GeoDjango [14].

4.2 Services

4.2.1 Get user authentication

After the user enters their username and password, the client will send a request to the server for authentication.

Input : Token (Username, Password)

Data Type : String, String

Output : Result

Data Type : Boolean

4.2.2 Tile map service

When the client sends an HTTP request containing the zoom level, x coordinate and y coordinate, the server side will return the image file.

Input : Zoom level, x coordinate and y coordinate

Data Type : Integer, Float and Float

Output : Image file

Data Type : PNG

4.2.3 Calculation service

In order to do a calculation, the client will send a table with object properties to the server side. The server side will do the calculation and return the result as a string.

Input : Table with object properties

Data Type : Two-dimensional array of type float

Output : Result table

Data Type : Two-dimensional array of type float

4.2.4 Search service

The client can enter search information as a string. Search information could be a pipeline ID and a power line ID. Then the server-side will return the list of geographical objects

for all the possible results.

Input : String

Data Type : String

Output : List of Geographical objects

Data Type : List

4.2.5 Get geographical object service

In order to load a map, a client needs to send a zoom level, an x coordinate, an y coordinate and a boundary as a string to the server-side and the server will return a geographical object corresponding to the data which is received from the client.

Input : Zoom level, x coordinate, y coordinate and current map boundary

Data Type : Integer, Float and Float

Output : Geographical object

Data Type : Geographical object

4.2.6 Create geographical object service

The user can create a new geographical object on the client-side, then send it to the server-side. The server-side will return the updated object to the client side.

Input : Zoom level, x coordinate, y coordinate and current map boundary

Data Type : Integer, Float and Float

Output : Geographical object

Data Type : Geographical object

4.2.7 Edit geographical object service

The user can edit a geographical object on the client-side, then send it to the server-side. The server-side will do the updating and return the geographical object to the client side.

Input : Zoom level, x coordinate, y coordinate and current map boundary

Data Type : Integer, Float and Float

Output : Geographical object

Data Type : Geographical object



Chapter 5

Development

5.1 GIS Data Formats

There are two main GIS data types which are raster data and vector data. But currently we use only the vector data type in the OSM format.

5.1.1 Vector data

Vector data represents real world features. These features include objects in a landscape. There are three types of objects:

Point

Point should be used to map objects that have a single location that can be marked by a single circle/symbol. This will include asset items such as a bin, bench, bus shelter, village pub, school or traffic sign [17].

Lines

Line should be used for linear objects that have a separate start and end point. This may include a fence line, walking route, section of river/stream, a section of road [17].

Polygon

Polygon should be used to mark out enclosed areas, that have a joined up boundary (perimeter). This may include the boundary of a Planning Application, Area of grass (for maintenance contracts), conservation areas, playgrounds, and allotments [17].

The information data of the map which show on leaflet on the client side are sent and received from the server side. Geojson file is used in order to exchange the data of the map between the client side and the server side.

5.2 Database

A database is a set of collection of data. In a relational database, data is organized into tables. We are using PostgreSQL with PostGIS extension to store our database.

5.2.1 Spatial database system

A spatial database system is a database system which is used to store, manage and access spatial data. It supports spatial data types and also supports a query language (QL) such as SQL. The data which is stored in a spatial database includes points, lines, polygons, and other geographic objects. To enable PostGIS, we need to execute the following query in PostgreSQL:

```
1 CREATE EXTENSION postgis;
```

5.2.2 Importing geospatial data into a spatial database

In order to import geospatial data from OpenStreetMap (OSM) into our spatial database, we use osm2pgsql, a program that runs in command-line to transfer data from an OSM file into a PostgreSQL database with PostGIS extension.

```
1 osm2pgsql saraburi.osm -d osm -U postgres -P 5432 -S default.style --hstore
```

The server administrator needs to use --append option switch in order to load additional data.


```
osm2pgsql saraburi.osm --append -d osm -U postgres -P 5432 -S default.style --hstore
```

If the data is loaded successfully, the server administrator will see four new tables each of which has a column which holds geographic object. By default, this column is called "way" and it stores the data in spherical web Mercator projection. The server administrator can specify a different projection.

The table called `planet_osm.point` contains points of interest such as restaurants, hospitals, schools, supermarkets and addresses.

The table called `planet_osm.line` contains rivers, railways, waterways, pipelines and power lines.

The table called `planet_osm.roads` contains roads and streets.

The table called `planet_osm.polygon` contains lakes, building footprints, administrative boundaries such as towns and cities.

Our main work is on `planet_osm.line` where all of pipelines and power lines are stored.

5.3 Geographical Object Service

Additional libraries which are GEOS, GDAL and PROJ.4 are necessary to enable geospatial support for Django.

Geometry Engine - Open Source (GEOS) is a C++ port of the Java Topology Suite (JTS). OpenGIS Simple Features are included in SQL spatial predicate functions and spatial operators.

Geospatial Data Abstraction Library (GDAL) is a library which is designed to read and write vector geographic data in the standard format. It also has a useful command line utility for both data translation and processing.

PROJ.4 is a geodetic computation and cartographic transformation library. It can convert longitude and latitude to x, y coordinates under the specified map projection.

We mainly use GEOS library because we can represent the spatial data from the database in GEOS geometry objects. They can be included in a Django model which can be used by

Django REST Framework to provide geographical object services. The following line of code is an example of how to declare GEOSgeometry.

```
1 geometry = models.GeometryField(db_column='way',null=True,srid=3857)
```

To filter a Django model that contain GEOSgeometry is similar to filter other common Django query sets. This is one of many ways to filter objects in Django views.

```
1 Line.objects.filter(Q(category='pipeline')|Q(category='powerline')).transform(srid=4326)
```

This method returns every geospatial object that is a power line or a pipeline. Other objects such as roads or houses are not included in the queryset. The coordinate system of the resulting objects is transformed from EPSG:3857 into EPSG:4326 because the client-side application uses coordinate data with EPSG:4326 which is easily readable by the client application and the user.

5.4 Tile Map Rendering

The server-side system uses Mapnik to generate tile map images from geographic data located in the database which will be used to serve the client-side application later. The first thing we need to do after we have finished Mapnik installation in the system is to generate a styling XML file. The XML file contains all the rules for generating tile map images by looking at the property fields in planet.osm database tables.

We use default Mapnik styling that is widely used by popular websites such as OpenStreetMap, so we can directly use generate_xml.py that is provided by the installation of Mapnik. We can generate this XML file by using this command.

```
1 generate_xml.py saraburi.xml /osm_tiles/ma_osm.xml
2 --dbname osm --user postgres --port 5432 --extent --accept-none
```

Mapnik also provides us generate_tiles.py for generating tile map images according to the specified zoom level and bounding box. Traditionally, we can schedule re-rendering of tile map images, for example, every 30 minutes. This usually takes a very long time as Mapnik itself does not know the change inside the database, so it will re-render every

object inside the database. To reduce re-rendering time, we write some code to enable re-rendering when there are some changes inside the database from the transactions processed by Django. Since both Django and Mapnik are both written in Python, we write python code inside model.py of the Django application to enable running re-rendering tile map images function after the client makes a change of data in the database. Here is the re-rendering function.

```
render_tiles(bbox, xml_file, tile_directory, 0, 18)
```

bbox is the bounding box which contains the minimum and maximum coordinates of the re-rendering area. xml_file is the location of the styling XML file generated from generate_xml.py tile_directory is the directory where the tile map images are located. The last two parameters are the minimum and maximum zoom levels of the generated maps. Normally, we have to write PostgreSQL queries in order to get the bounding box of the new or modified geographical object in the database. But thanks to GEOS, we can access the bounding box directly which is included in the GEOSgeometry object in every Django model in our python code without having to write any SQL code directly in the system.

5.5 Electrical Interference Calculation

For the calculation of electrical interference on pipelines, we need the formula that we described in our project description to do the calculation. The data we need is from the properties of pipelines and power lines in the database such as the complex value of the transmission rate of a pipeline. There is also other data in JSON format that we get when the client-side application makes a calculation request. All the mathematical formulas which we use to calculate in this project are from the TE7 document [2]. We use JSON format for data exchange. The data exchange is performed by Django Rest framework.

5.6 User Authentication and Registration System

If the Django admin site is included within the installed app in the Django settings, it will enable the authentication system. It allows the system administrator to handle user

accounts through a web browser. There are 3 types of users in the system. Unregistered users are not allowed to use any of the geographical object services. Registered users can view geographical objects but are not allowed to edit them. An analyzer can view and edit geographical objects without restrictions. We have to write a python class to customize user permissions. Here is the example source code.

```
1 class PipeabsolutePermission(permissions.BasePermission):
2
3     def has_permission(self, request, view):
4         if request.user:
5             if request.user.is_staff:
6                 return True
7             elif request.user.is_authenticated:
8                 return (request.method in ('GET', 'HEAD', 'OPTIONS'))
9         return False
```

5.6.1 Registration Service

Django REST framework handles most of the registration process. A user can register using the client-side application by filling in personal information such as their username and e-mail address. After the user has filled in the registration form, the client will send the information to server through Django REST services. Then the server-side system itself will generate a password and send the generated password to user by e-mail. We use the Django.core.mail module which comes with the installation of Django to handle the mail sending process. The Django mail module sends mails through the simple mail transfer protocol (SMTP). We use a specific SMTP server that we set in the setting file of the Django application. For the SMTP server, we use the Google SMTP server because when the system sends an e-mail, it will not be marked as a spam.

5.6.2 Authentication of Services

In order to create a fully REST service without needing to keep user sessions in the server-side storage, we implement user authentication by using JWT tokens. The Django rest framework plugin called Rest framework JWT Auth is used to handle JWT authentication. The first step in the authentication happens when a client user types their username and

password on the login page of the client-side application. The client system will then send the username and the password to the server-side system. Then the system will generate a JWT token and send this token back to the client-side application to be used to request an access to other secured services such as geographical object service. The system will check if the token is valid or not before granting the client access to each service.



Chapter 6

Results

6.1 Tile Map Service

Our system follows the Tile Map Service protocol for Slippy Map Web Applications for serving maps as tiles in different zoom levels and coordinates. All map images that are generated from Mapnik are always 256x256 pixel PNG files. All images are stored as cache files located on the hard drive where our system is deployed, so it speeds up the retrieval the next time the client-side application requests the map because the rendered portion of the map is not needed to be rendered again. Map images are re-rendered only when the pipeline or the power line data in the database has been changed. This is done using Django REST services in the system.

The client - Side application can request tile map service using the URL:

.../tiles/z/x/y.png

where z is the zoom level, x is the x-coordinate and y is the y-coordinate.

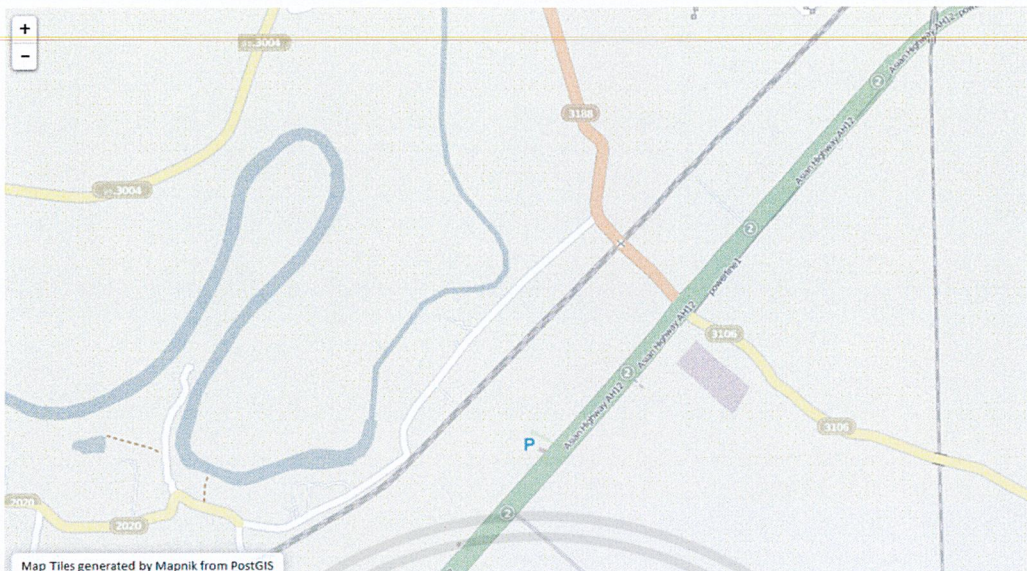


Figure 6-1: A map generated by Mapnik

6.2 Registration

The user can register for a new account using the client-side application. After the user provides their personal information on the registration page, the client application will send a POST request in JSON format to the server-side system.

```
1 {  
2   "username": "yosbutter",  
3   "email": "buttercaked@hotmail.com",  
4   "first_name": "Yosavich",  
5   "last_name": "Daolomchan",  
6   "country": "Thailand"  
7 }
```

After the server has received the request, the server will validate the data. If the data is valid, the server will generate a random password and then sends the password to the email of that registered user.

The administrator of the system can later change the status and other information of the user on the Django admin site.

< All Inboxes (104) >
FROM: npolawat@...
To: Y Y > Hide

Welcome to PipeAbsolute
Today at 13:24

This is your password
HqXQb0UB

Figure 6-2: Password is sent via e-mail

Change user

Username: yosbutter
Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only.
Password: algorithm: pbkdf2_sha256 iterations: 36000 salt: 1Y406Q***** hash: Sic2Lh*****
Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using the form.

Personal info

First name: Yosavich
Last name: Daolomchan
Email address: buttercaked@hotmail.com

Permissions

☒ Active
Designates whether this user should be treated as active. Unselect this instead of deleting accounts.
☒ Staff status
Designates whether the user can log into this admin site.
☐ Superuser status
Designates that this user has all permissions without explicitly assigning them.

Figure 6-3: Administrator can change status of user

6.4.1 Sample of the description of a power line in the GeoJSON format

```
1  {
2  "id": 13976,
3  "type": "Feature",
4  "geometry": {
5    "type": "LineString",
6    "coordinates": [
7      [
8        100.97253648800006,
9        14.559363244690006
10     ],
11     [
12       100.98084420077002,
13       14.568847574639992
14     ],
15     [
16       100.98088848408001,
17       14.569029167829983
18     ],
19     [
20       101.08419117778004,
21       14.616404447749963
22     ]
23   ]
24 },
25 "properties": {
26   "name": "Powerline_test",
27   "category": "Powerline",
28   "slug": "powerline_test",
29   "timestamp": "2018-04-09T20:41:00Z",
30   "object_type": "Double Line",
31   "operator": "PEA",
32   "voltage": "380000.0000000000",
33   "height": "30.0000000000",
34   "frequency": "50.0000000000",
35   "ground_resistance": "100.0000000000",
36   "load": "",
37   "details": "http://10.94.15.74:83/geojson/powerline/13976/"
38 }
39 }
40 }
```

6.4.2 Sample of the description of a segmented line in the GeoJSON format

```
1  {
2  "id": 3,
3  "type": "Feature",
4  "geometry": {
```



```

5      "type": "LineString",
6      "coordinates": [
7          [
8              100.759047700000004,
9              14.562930000000005
10         ],
11         [
12             100.98016552632002,
13             14.579876877689994
14         ]
15     ]
16 },
17 "properties": {
18     "details": "http://10.94.15.74:83/geojson/section/3/",
19     "name": "section-test",
20     "slug": "abrwg",
21     "area": "A1",
22     "type_of_segment": "Earthfault",
23     "pipeline": 13984,
24     "powerline": 13976
25 }
26 }
```

6.4.3 Sample of the description of a pipeline in the GeoJSON format

```

1      {
2      "id": 13984,
3      "type": "Feature",
4      "geometry": {
5          "type": "LineString",
6          "coordinates": [
7              [
8                  100.759047700000002,
9                  14.562930000000005
10             ],
11             [
12                 100.98016552632002,
13                 14.579876877689994
14             ],
15             [
16                 101.01150921631,
17                 14.583436125129975
18             ],
19             [
20                 101.01455802095005,
21                 14.584656084569978
22             ],
23             [
24                 101.03279947838003,
25                 14.592496406490007
26             ]
27         ]
28     }
29 }
```



```

27     [
28         101.05221513670003,
29         14.59640902477
30     ],
31     [
32         101.05330538909,
33         14.596646735619974
34     ],
35     [
36         101.06377292117004,
37         14.598701131559991
38     ],
39     [
40         101.183395000000005,
41         14.612820799999997
42     ]
43 ],
44 },
45 "properties": {
46     "name": "pipeline-test",
47     "category": "Pipeline",
48     "slug": "pipeline-test",
49     "timestamp": null,
50     "object_type": "DN 500",
51     "thickness": "3.3000000000",
52     "diameter": "500.0000000000",
53     "resistance_of_coating": "1.0800000000",
54     "depth": "1.5000000000",
55     "material_of_coating": "Polyurethane",
56     "ground_type": "",
57     "grounding_point": "16.0000000000",
58     "transmission_rate": "0.1360000000",
59     "details": "http://10.94.15.74:83/geojson/pipeline/13984/"
60 }
61 }

```

6.5 Calculation

Here is the sample of the data in JSON format that we receive from the client-side application in order to do the electrical interference calculation. The client sends each section with its information and also both pipeline IDs and power line IDs. These IDs are needed for the calculation because we use them to identify the pipeline and the power line objects in our database which the system need their properties, such as transmission rate and voltage, in order to calculate the results.


```

1 {
2   "json_data" : {
3     "1-2": {
4       "section_pipeline_id": 157,
5       "section_powerline_id": 10
6     }
7   }
8 }

```

The system calculates the result according to the formula provided by TE7 document[2]. The code below shows the result in JSON format which will be sent back to the client-side application.

```

1 {
2   "result": {
3     "1-2": {
4       "a": 822.0,
5       "column9": 14.1,
6       "column11": 17.7,
7       "column10": 17.7,
8       "LH": 1.213,
9       "a1": 1000.0,
10      "a2": 1000.0,
11      "Sum_LR": 1.26,
12      "column5": 14.6,
13      "LR": 1.256
14    }
15  }
16 }

```


Chapter 7

Conclusion

7.1 Project Summary

GIS is a useful tool but it challenges us because we did not have much knowledge about it. However after doing much research we have learnt many useful techniques and new skills for developing a server using PostgreSQL with the PostGIS extension in order to access and manage spatial data in a database. In our project we have developed a server-side system for the analysis of AC interference on pipelines, which is a web application for analyzing interference across a pipeline system running along an overhead power line.

We use Django to act as a middleware between the database and the client-side application. We use the Django REST framework to implement data transfer services that are restful. Mapnik is used for map generation from the geospatial database which is cached in the hard drive where our system is located and later used to create a tile map service.

All the requirements of the server-side system for the analysis of AC interference on pipelines have been fulfilled.

7.2 Problems and Obstacles

While we implemented and developed a web application for the analysis of AC interference on pipelines there were three main problems that occurred.

The first problem was that in this project we needed electrical engineering knowledge in order to understand formulas to know which variable was use for and whether it was a user input or came from some calculation. It also took us a long time due to language barrier because all the documents were written in German language. DVGW GW 22 (A) and THE-7 SfB 2014 [2] were the documents that were provided in order to analyze AC interference on pipelines and we could not translate them directly for the reason that both of these documents were written by electrical engineers so most content was written in technical language. However, we received help from Dipl.Ing Michael Juelich for explanation and translation. He also taught us about electrical engineering knowledge which is needed in order to do the analysis AC interference on pipelines.

The second problem was the difficulty in connecting with the client-side which took us a long time to find a solution but in the end we fixed it by adding the Django module called CORS header.

The last problem was that we did not have experience with Django, so it had appeared to be difficult to us at the beginning but in the end Django was not that difficult to use. It was just that there were many tiny details. There were many documents and also sample code for us to learn.

7.3 Future Work

We have finished all the required features but there are possible future work that can be added to the project.

- Optimization of map rendering. In our work, we did not study much about map generation algorithms for processing an image generated by Mapnik. When there are concurrent data changes in the system, the map rendering in a high zoom level is still slow.
- Improving the system to allow a large number of users to access the system at the same time.
- Deploying and maintenance of the entire system in the long term.

- Improving the security of the server-side system. There might be still some security holes which allow hackers to gain unauthorized access to the system.



Bibliography

- [1] G. Jacobson, "Corrosion Basics", Nace.org. [Online]. Available: <https://www.nace.org/Corrosion-Basics>. [Accessed: 27- Feb- 2017].
- [2] DVGW GW 22 -Arbeitsblatt, 1st ed. Bonn: DVGW German Association of Gas and Water Sinks Technical-Scientific Association, 2014.
- [3] "protective Lining", Svobodny Sokol, [Online]. Available: <http://www.svsokol.pl/en/protective-coating-and-lining> [Accessed: 17 - Oct - 2017]
- [4] "Cathodic protection", Corrosion source, [Online]. Available: <http://www.corrosionsource.com/FreeContent/1/Cathodic+Protection> [Accessed: 17 - Oct - 2017]
- [5] "Pipeline AC Interference and Mitigation Overview", matcor.com [Online] Available: <http://www.matcor.com/resources/education/pipeline-ac-interference-mitigation-overview/> [Accessed: 20 - Oct- 2017].
- [6] "Corrosion Inhibitors", World News. [Online]. Available: https://wn.com/corrosion_inhibitors. [Accessed: 19- Oct- 2017].
- [7] "What is a Pipeline Coating", Corrosionpedia. [Online]. Available: <https://www.corrosionpedia.com/definition/1324/pipeline-coating>. [Accessed: 16- Oct- 2017].
- [8] "Point Integrity Solution", pointintegrity. [Online]. Available: <http://www.pointintegrity.com/m/home>. [Accessed: 17- Oct- 2017].

- [9] S. Prabhu, "An Intro to Pipeline Corrosion and Coatings", Corrosionpedia, 2016. [Online]. Available: <https://www.corrosionpedia.com/2/1383/corrosion-101/an-intro-to-pipeline-corrosion>. [Accessed: 27- Oct- 2017].
- [10] "Introducing JSON", json.org. [Online]. Available: <https://www.json.org/>. [Accessed: 11- Dec- 2017].
- [11] "Coatings and Linings for Pipelines", Masterbond.com. [Online]. Available: <http://www.masterbond.com/industries/coatings-and-linings-pipelines>. [Accessed: 28- Feb- 2017].
- [12] "What is a GIS", Rst2.edu, 1998. [Online]. Available: http://www.rst2.edu/ties/gentools/what_gis.html. [Accessed: 14- May- 2017].
- [13] C. Sue, "GIS (geographic information system)", National Geographic Society, 2011. [Online]. Available: <https://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/>. [Accessed: 14- May- 2017].
- [14] O. Obe, S. Hsu, "PostGIS in Action, Second Edition".
- [15] <https://docs.djangoproject.com/en/2.0/ref/contrib/gis/tutorial/introduction>. [Accessed: 12- OCT- 2017].
- [16] <https://github.com/mapnik/mapnik/wiki/About-Mapnik>. [Accessed: 14- OCT- 2017].
- [17] "The difference between point, line and polygon", parishonline.geoxphere.com [Online] Available: <http://parishonline.geoxphere.com/component/content/article/28-support-article/55-what-s-the-difference-between-point-line-and-polygon-information?Itemid=308> [Accessed: 20 - April- 2018].